Interrupts
Day 2 of 2

Panel 1

Prior to Le15

Interrupts

Panel 2

Interrupt Priority Feature

<table>
<thead>
<tr>
<th>Mode</th>
<th>Normal</th>
<th>All Intros High Priority</th>
<th>High and Low Intros</th>
</tr>
</thead>
<tbody>
<tr>
<td>All Intros</td>
<td>On</td>
<td>Off</td>
<td>On</td>
</tr>
<tr>
<td>High Priority</td>
<td>Off</td>
<td>High Priority on</td>
<td></td>
</tr>
</tbody>
</table>

Global / High Priority Bit

<table>
<thead>
<tr>
<th>Interrupt</th>
<th>Mode</th>
<th>Require</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>All Intros</td>
<td>Off</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>High Priority</td>
<td>Off</td>
<td>-</td>
<td>Off</td>
</tr>
</tbody>
</table>

Peripheral / Low Priority Bit

<table>
<thead>
<tr>
<th>Interrupt</th>
<th>Mode</th>
<th>Require</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Peripheral</td>
<td>Off</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Low Priority</td>
<td>Off</td>
<td>-</td>
<td>Off</td>
</tr>
</tbody>
</table>

Interrupt on Change

- On RB4, RB5, RB6, RB7 (KB0, KB1, KB2, KB3)

<table>
<thead>
<tr>
<th>Enable</th>
<th>Mode</th>
<th>Require</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>RB4, RB5, RB6, RB7</td>
<td>Off</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Panel 3

Interrupt 0 - On RB0 (INT0)

<table>
<thead>
<tr>
<th>Enable</th>
<th>Mode</th>
<th>Require</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTCONbits.INT0IE</td>
<td>Off</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>INTCONbits.INT0IF</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>INTCON2bits.INTEDG0</td>
<td>Falling Edge</td>
<td>Rising Edge</td>
<td></td>
</tr>
</tbody>
</table>

Interrupt 1 - On RB1 (INT1)

<table>
<thead>
<tr>
<th>Enable</th>
<th>Mode</th>
<th>Require</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTCONbits.INT1IE</td>
<td>Off</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>INTCONbits.INT1IF</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>INTCON2bits.INTEDG1</td>
<td>Falling Edge</td>
<td>Rising Edge</td>
<td></td>
</tr>
</tbody>
</table>

Interrupt 2 - On RB2 (INT2)

<table>
<thead>
<tr>
<th>Enable</th>
<th>Mode</th>
<th>Require</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTCONbits.INT2IE</td>
<td>Off</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>INTCONbits.INT2IF</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>INTCON2bits.INTEDG2</td>
<td>Falling Edge</td>
<td>Rising Edge</td>
<td></td>
</tr>
</tbody>
</table>

Panel 4

Timer 0 Interrupt

<table>
<thead>
<tr>
<th>Enable</th>
<th>Mode</th>
<th>Require</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTCONbits.TMR0IE</td>
<td>Off</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>INTCONbits.TMR0IF</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>INTCON2bits.TMR0IP</td>
<td>Low Priority</td>
<td>High Priority</td>
<td></td>
</tr>
</tbody>
</table>

Timer 1 Interrupt

<table>
<thead>
<tr>
<th>Enable</th>
<th>Mode</th>
<th>Require</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIEbits.TMR1IE</td>
<td>Off</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>PIRbits.TMR1IF</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>IPRbits.TMR1IP</td>
<td>Low Priority</td>
<td>High Priority</td>
<td></td>
</tr>
</tbody>
</table>

Timer 2 Interrupt

<table>
<thead>
<tr>
<th>Enable</th>
<th>Mode</th>
<th>Require</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIEbits.TMR2IE</td>
<td>Off</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>PIRbits.TMR2IF</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>IPRbits.TMR2IP</td>
<td>Low Priority</td>
<td>High Priority</td>
<td></td>
</tr>
</tbody>
</table>

Timer 3 Interrupt

<table>
<thead>
<tr>
<th>Enable</th>
<th>Mode</th>
<th>Require</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIE2bits.TMR3IE</td>
<td>Off</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>PIR2bits.TMR3IF</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>IPR2bits.TMR3IP</td>
<td>Low Priority</td>
<td>High Priority</td>
<td></td>
</tr>
</tbody>
</table>
2.5 I/O PORT FUNCTIONS

PORTB is supported with the following functions:

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ClosePORTB</td>
<td>Disable the interrupts and internal pull-up resistors for PORTB.</td>
</tr>
<tr>
<td>Close2RxINT</td>
<td>Disable interrupts for PORTB pin x.</td>
</tr>
<tr>
<td>DisablePullups</td>
<td>Disable the internal pull-up resistors on PORTB.</td>
</tr>
<tr>
<td>EnablePullups</td>
<td>Enable the internal pull-up resistors on PORTB.</td>
</tr>
<tr>
<td>OpenPORTB</td>
<td>Configure the interrupts and internal pull-up resistors on PORTB.</td>
</tr>
<tr>
<td>OpenRxINT</td>
<td>Enable interrupts for PORTB pin x.</td>
</tr>
</tbody>
</table>

Only help you set the individual enable bits. Don’t help with global setting (SIE or IPEN) Don’t help with flag within ISR.

---

The PORTB Interrupt on change function

**OpenPORTB**

- **Function:** Configure the interrupts and internal pull-up resistors on PORTB.  
- **Include:** portb.h  
- **Prototype:** void OpenPORTB( unsigned char config );  
- **Arguments:** config  
  A bitmask that is created by performing a bitwise AND operation (‘&’) with a value from each of the categories listed below. These values are defined in the file portb.h.  
- **Interrupt-on-change:**  
  - PORTS_CHANGE_INT_ON: Interrupt enabled  
  - PORTS_CHANGE_INT_OFF: Interrupt disabled  
- **Enable Pullups:**  
  - PORTS_PULLUPS_ON: Pull-up resistors enabled  
  - PORTS_PULLUPS_OFF: Pull-up resistors disabled  
- **Remarks:** This function configures the interrupts and internal pull-up resistors on PORTB.  
- **File Name:** portopen.c  
- **Code Example:** OpenPORTB( PORTS_CHANGE_INT_ON & PORTS_PULLUPS_ON );

---

Timer library function to help set that interrupt

**OpenTimer0**

- **Function:** Configure and enable timer0.  
- **Include:** timers.h  
- **Prototype:** void OpenTimer0( unsigned char config );  
- **Arguments:** config  
  A bitmask that is created by performing a bitwise AND operation (‘&’) with a value from each of the categories listed below. These values are defined in the file timer0.h.  
- **Enable Timer0 interrupt:**  
  - TIMER_INT_ON: Interrupt enabled  
  - TIMER_INT_OFF: Interrupt disabled  
- **Remarks:** This function configures the interrupts and internal pull-up resistors on PORTB.  
- **File Name:** rbtopen.c  
- **Code Example:** OpenTimer0( PORTS_CHANGE_INT_ON & PORTS_PULLUPS_ON & RISING_EDGE_INT & PORTS_PULLUPS_ON );
Let's do a big crazy example to make sure you get the idea with interrupts!

Let's make a program with 3 interrupts using the Priority Mode
(That's about as hard as we'll make any interrupt problems)

Timer 3 interrupt
- Low priority
- 16 bit timer with 1:4 Prescaler

INT2 interrupt
- High priority
- Rising edge

PORTB RB4:RB7 interrupt on change
- Low priority

What should this crazy program do?

/** Global Variables **********
int RBinterrupts = 0;
int timeoverflows = 0;
int resets = 0;
int RBmagnitude = 0;

Timer 3 interrupt
- Increments timerOverflows

INT2 interrupt
- Resets RBinterrupts, timerOverflows, and RBmagnitude
- Increments resets

PORTB RB4:RB7 interrupt on change
- Increases RBmagnitude based on PORTB
- Increments RBinterrupts

Start by making a new project using the "template with interrupts.c"
(located on website under Courseware)

void main(void)
{
    // Setup pins to be digital
    // Setup PORTB to be inputs
    // Put the interrupts into Priority Mode
    // Turn on the RB2 interrupt INT2 use the library
    // Make it rising edge and high priority
    // Turn on the Change on RB4:RB7 interrupt
    // Make it low priority
    // Start up Timer 3 with low priority interrupts
    // Use bit mode with a 1:4 Prescaler
    // Turn on High Priority interrupts
    // Turn on Low Priority interrupts
    while (1)
    {
        // This area loops forever but does nothing at all
    }
}

Start trying to write the code to perform the comments

// Setup pins to be digital
ADCON1 = 0x0F;

// Setup PORTB to be inputs
TRISB = 0xFF;

...
Panel 13

Copy in the comments for the High Interrupt Service Routine

/**
 * Function:   void high_isr(void)
 * Purpose:    
 * #pragma interrupt high_isr
 void high_isr(void)
 {
  // High Priority Interrupt Service Routine (High ISR)
  // See if it was due to Interrupt 2 (it should be)
  // Reset the flag and counters, increment resets
 }
*/

Try to write the code for these

Panel 14

Copy in the comments for the Low Interrupt Service Routine

/**
 * Function:   void low_isr(void)
 * Purpose:    
 * #pragma interrupt low_isr
 void low_isr(void)
 {
  // Low Priority Interrupt Service Routine (Low ISR)
  // See if it's due to the timer overflow
  // If it is reset the flag and increment timeOverflows
  // See if it's due the change on RB4/RB7
  // If it is increase xMaxamplitude, increment XInterrupts, reset flag
  // Warnings: Can only clear this flag AFTER reading PORTB
 }

Try to write the code for these

Panel 15

Add the Watch window and Stimulus

Panel 16

/**
 * Header Files 
 ***/
#include <p8sf4520.h>
#include <portb.h>
#include <time.h>

/**
 * Global Variables 
 ***/
int XInterrupts = 0;
int timeOverflows = 0;
int resets = 0;
int xMaxamplitude = 0;

// Setup pins to be digital
ADCON1 = 0x0F;

// Setup PORTB to be inputs
TRISB = 0x0F;
// Put the interrupts into Priority Mode
#include <avr/interrupt.h>

// Turn on the R82 interrupt INT2 use the library
// Make it rising edge and high priority
IOCONbits.INT2 = RISING_EDGES_INT & PORTB_PULLUP_OFF;

// Turn on the Change on R84/R87 interrupt
// Make it low priority
IOCONbits.INT2 = PORTB_PULLUP_OFF;

// Start up Timer 3 with low priority interrupts
// Use bit mode with a 1:1 Prescaler
PORTPbits.T3M3IP = T3_1MTH_DIV & T3_SOURCE_INT & T3_PS_1_6;

// Turn on High Priority interrupts
IOCONbits.GIE = 1;

// Turn on Low Priority interrupts
IOCONbits.GIEL = 1;

/**
 * Function: void high_isr()
 * Purpose: 
 */

void high_isr(void)
{
    // High Priority Interrupt Service Routine (High ISR)
    // Reset if it was due to Interrupt 2 (it should be)
    // Reset the flag and counters, increment resets
    if (INTCONbits.INT2)
    {
        INTCONbits.INT2 = 0;
        R8interrupts = 0;
        timeOverflows = 0;
        R8magnitude = 0;
        resets++;
    }
}

/**
 * Function: void low_isr()
 * Purpose: 
 */

void low_isr(void)
{
    // Low Priority Interrupt Service Routine (Low ISR)
    // See if it's due to the timer overflow
    // If it is reset the flag and increment timeOverflows
    if (PORTPbits.T3M3IP)
    {
        PORTPbits.T3M3IP = 0;
        timeOverflows++;
    }

    // See if it's due the change on R84/R87?
    // If it is increases R8magnitude, increment R8interrupts, reset flag
    // Watch note: Can only clear this flag AFTER reading PORTS
    if (INTCONbits.INT2)
    {
        if (PORTBbits.RB1)
        {
            R8magnitude = R8magnitude + 3;
            if (PORTBbits.RB2)
                R8magnitude = R8magnitude + 2;
            if (PORTBbits.RB1)
                R8magnitude = R8magnitude + 1;
        }
        else
        {
            R8magnitude = R8magnitude + 0;
            R8interrupts++;
        }
        IOCONbits.RBIF = 0;
    }
}